

Hyperparameters optimization XGBoost for network intrusion detection using CSE-CIC-IDS 2018 dataset

Witcha Chimphee, Siriporn Chimphee

Department of Data Science and Analytics, Faculty of Science and Technology, Suan Dusit University, Bangkok, Thailand

Article Info

Article history:

Received Jun 2, 2023

Revised Oct 18, 2023

Accepted Oct 21, 2023

Keywords:

Extreme gradient boosting
Hyperparameters
Machine learning
Network intrusion detection
XGBoost

ABSTRACT

With the introduction of high-speed internet access, the demand for security and dependable networks has grown. In recent years, network attacks have gotten more complex and intense, making security a vital component of organizational information systems. Network intrusion detection systems (NIDS) have become an essential detection technology to protect data integrity and system availability against such attacks. NIDS is one of the most well-known areas of machine learning software in the security field, with machine learning algorithms constantly being developed to improve performance. This research focuses on detecting abnormalities in societal infiltration using the hyperparameters optimization XGBoost (HO-XGB) algorithm with the Communications Security Establishment-The Canadian Institute for Cybersecurity-Intrusion Detection System2018 (CSE-CIC-IDS2018) dataset to get the best potential results. When compared to typical machine learning methods published in the literature, HO-XGB outperforms them. The study shows that XGBoost outperforms other detection algorithms. We refined the HO-XGB model's hyperparameters, which included `learning_rate`, `subsample`, `max_leaves`, `max_depth`, `gamma`, `colsample_bytree`, `min_child_weight`, `n_estimators`, `max_depth`, and `reg_alpha`. The experimental findings reveal that HO-XGB1 outperforms multiple parameter settings for intrusion detection, effectively optimizing XGBoost's hyperparameters.

This is an open access article under the [CC BY-SA](#) license.



Corresponding Author:

Siriporn Chimphee

Department of Data Science and Analytics, Faculty of Science and Technology, Suan Dusit University
Bangkok, Thailand

Email: siriporn_chi@dusit.ac.th

1. INTRODUCTION

As the internet becomes more widely used, an increasing number of computers are being networked. However, with the rapid advancement of digital technology, network data traffic has become vulnerable to numerous security risks and potential breaches. A vast amount of information has been transmitted across intricate network connections worldwide. Consequently, the establishment of a secure information system has garnered significant attention from both private and governmental institutions, aiming to thwart potential attackers [1]. Network attacks have emerged as one of contemporary society [2]. Intrusions refer to actions aimed at bypassing the security measures of computer systems [3]. The complexity and challenge of intrusion detection systems in heterogeneous networks are heavily influenced by the variety of devices, protocols, and services employed. As a result, the network's intricacy increases, making it arduous to identify and detect intrusions [4]. To address this pressing need for stronger protection, intrusion detection has become crucial in monitoring computer systems and networks, and analyzing events for signs of potential intrusions [5].

The intrusion detection system (IDS) is a dynamic monitoring system that assesses system activity in a given environment to determine whether it constitutes an attack [6], [7]. IDS systems can be categorized into two types based on their detection methods: signature-based and anomaly-based. Signature-based IDS functions similarly to antivirus software by identifying known attack patterns or signatures. Although it has high accuracy and a low rate of false positives for known attacks, it lacks the ability to detect new attacks. This type of IDS generally has a higher false-positive rate, and building a model requires a reliable training dataset.

Anomaly-based IDS is cybersecurity solutions that monitor network traffic, system activity, or user behavior to detect deviations from set baselines. They detect odd patterns or suspicious actions by utilizing machine learning and statistical methodologies, potentially identifying unforeseen risks. However, they may generate false positives and require ongoing calibration to reduce such warnings [8]-[12]. A network intrusion detection system (NIDS) is a critical component in identifying ongoing attacks by differentiating normal network traffic from malicious traffic [13]. NIDS is essential in resolving security issues by monitoring network traffic for potential signs of suspicious activity and detecting any security vulnerabilities, such as infiltration, abuse, and anomalies, in the data extracted from network traffic [14].

Traditional intrusion detection systems often use machine learning-based methods, and many different machine learning algorithms have been developed and are available for use [15]-[19]. Machine learning algorithms that are widely accepted and being used include: 1) logistic regression (LR), 2) decision tree, 3) random forest, 4) extreme gradient boosting (XGBoost), and 5) k-nearest neighbor. These algorithms are a classification models that predicts the target class of data samples in classification problems. The main problem is the necessity to identify fresh, complex attack patterns that traditional intrusion detection systems frequently overlook and which pose a serious risk to network security. In order to prevent overtaxing security staff and ineffective incident response, it is critical to address the issue of a high false-positive rate concurrently. Therefore, the main research issue is to develop an intrusion detection system that can accurately identify new attack patterns while also lowering false alarms. To achieve this goal, the IDS's accuracy and responsiveness in fending off developing cyber threats must be improved. To do this, enhanced anomaly detection techniques, machine learning algorithms, and the integration of realtime threat intelligence must be investigated.

The study's goal is to create an effective intrusion detection system capable of distinguishing between normal network traffic and malicious activity and quickly detecting potential security weaknesses such as infiltration, misuse, and anomalies. Hyperparameter optimization is a process that involves fine-tuning the input parameters, or hyperparameters, that affect a machine learning algorithm's training phase and model. Because of the scale of the challenge, hyperparameter adjustment is frequently required in machine learning jobs [16]. As a result, tuning hyperparameters will be critical for achieving peak performance in machine learning tasks. This research focuses on intrusion detection, which is critical in monitoring computer systems and networks for signals of potential security breaches. We are specifically concerned with the issues posed by intrusion detection systems in heterogeneous networks, where the broad assortment of devices, protocols, and services makes reliably identifying and detecting intrusions extremely challenging.

2. BACKGROUNDS AND RELATED WORKS

Because of the complexity and diversity of today's networks, intrusion detection is a difficult task. Traditional intrusion detection systems frequently employ machine learning-based methods, but the number of available algorithms, as well as the requirement for effective hyperparameter tweaking, pose considerable obstacles. The key research question addressed in this paper is: How can we create an effective IDS that can consistently discriminate between normal network traffic and hostile activities while rapidly detecting potential security flaws? Our primary goal is to develop and deploy an IDS that uses machine learning techniques to accurately classify network traffic. In addition, we intend to undertake a thorough performance evaluation of five different machine learning algorithms (LR, decision tree, random forest, XGBoost, and k-neighbor) in the context of intrusion detection. Each machine learning algorithm has its own set of benefits and drawbacks, which are discussed more below. LR is a simple and uncomplicated technique that works effectively when the connection between features and the target variable is linear. It delivers probabilities for binary classification tasks, has a low computing overhead, and can quickly train large datasets. LR, on the other hand, has difficulties when it comes to capturing complex correlations between features and the target variable, and it tends to underperform when the data is not linearly separable. It is also susceptible to outliers and multicollinearity. LR must be modified utilizing one-vs-all or one-vs-one techniques to be relevant in circumstances involving several classes [20].

The simplicity of the decision tree, which provides unambiguous decision rules that are easy to comprehend and follow, is one of its many advantages. It is capable of handling both numerical and

categorical data without the need for considerable data preprocessing. It is resilient to outliers and does not presume a certain data distribution because it is non-parametric. Furthermore, decision trees can recognize non-linear correlations in data. However, there are some disadvantages to consider. Decision trees have a tendency to overfit the training data, especially as the tree grows in depth. They are extremely sensitive to slight changes in the data, resulting in unstable trees. Biased trees may be formed if some classes dominate the data, and they may be limited in their capacity to generalize well to unknown data, particularly in complicated datasets [21].

Random forest has various advantages over individual decision trees, including enhanced accuracy due to less overfitting using ensemble approaches. Because it aggregates several trees, it is resistant to outliers and noisy data, and it can handle high-dimensional data with numerous characteristics. Random forest also allows for the rating of feature relevance, which benefits in feature selection. There are, however, certain disadvantages to consider. When compared to individual decision trees, it has a higher level of complexity and processing expense. Random forest is opaque, making it difficult to understand the basis behind forecasts. Furthermore, its prediction time is longer than that of single decision trees, and characteristics such as the number of trees must be tuned for improved results [20], [22].

K-nearest neighbor (KNN) has a number of advantages, including its simplicity and intuitive nature, which makes it simple to apply. It is well-suited for capturing non-linear relationships in data because it is a non-parametric technique. KNN does not need a training phase because it memorizes the data points during training. It is effective with small to medium-sized datasets. However, there are some disadvantages to consider. Because distances to all data points must be determined, the testing process might be computationally expensive. The distance metric chosen can have a considerable impact on its performance. KNN is extremely sensitive to the existence of irrelevant features, which could result in bad outcomes. To avoid biased results, proper data standardization is essential [23].

Extreme gradient boosting, or XGBoost, is a supervised learning technique that belongs to the family of gradient-boosted decision trees (GBDT) machine learning algorithms. It was created by Chen and Guestrin [24] for classification and regression problems. XGBoost is trained by iteratively adding based learners in the form of decision trees to an ensemble while minimizing a regularized objective function. In XGBoost, the objective is to minimize the difference between the predicted values $(p_i)^{(t-1)}$ and the actual values (y_i) using a loss function. To do this, the algorithm adds decision trees (f_c) to the ensemble iteratively, with each tree decreasing the difference in predictions from the preceding iteration. To prevent overfitting, the regularization term penalizes the complexity of the extra trees. XGBoost is a powerful algorithm that is gaining increasing attention due to its speed and accuracy [25]. It is excellent at handling missing data, provides perceptions into the significance of features, and integrates regularization to avoid overfitting. It also scales effectively through parallel processing, making it appropriate for big datasets. XGBoost has a number of advantages, including superior prediction performance due to its boosting method, which concentrates on misclassified data points. It uses regularization techniques to avoid overfitting and is capable of processing a wide range of data types, including numerical and categorical data. Because of its parallelizable and efficient implementation, XGBoost is appropriate for huge datasets. There are, however, certain disadvantages to consider. If the hyperparameters are not appropriately set, it may be prone to overfitting. Furthermore, XGBoost might be difficult to read, especially when dealing with a large number of trees. When compared to other algorithms, training time can be longer, and attaining optimal results requires careful parameter optimization [2], [22].

Hyperparameters are pre-learning parameters that are not part of the model. Proper tuning of hyperparameters is critical for enhancing model performance and minimizing loss. The values of hyperparameters dictate the model parameters, and the purpose of hyperparameter tuning is to discover the best values that lead to optimal model performance and superior outputs. The regularization and construction of XGBoost are highly influenced by hyperparameters such as learning rate, ensemble size, and maximum depth of base learners. The goal of hyperparameter optimization in HO-XGB is to minimize the objective function in (1). Optimization of hyperparameters is an important task in automated machine learning since it improves model performance. But careful tuning of its many hyperparameters is required, which calls for a thorough knowledge of the method. It can be computationally and memory-intensive, especially when working with large datasets or deep trees. It differs from simpler models in interpretability as a black-box model that is a little less easy to understand. Managing unbalanced datasets effectively may also call for the use of extra methods or specifications.

$$\mathcal{L}^{(t)} = \sum_{i=1}^n l(y_i, p_i^{(t-1)}) + f_c(x_i) + \kappa(f_c) \quad (1)$$

Several hyperparameters are adjusted in this study, including `learning_rate`, `subsample`, `max_leaves`, `max_depth`, `gamma`, `colsample_bytree`, `min_child_weight`, `n_estimators`, `max_depth`, and `reg_alpha`. These

variables have a significant impact on the model architecture utilized by XGBoost [24], [26]. Although XGBoost has many hyperparameters in Table 1, this study only focuses on those that have been shown to significantly impact model performance in previous studies. The hyperparameters "subsample," "learning_rate," "max_leaves," "gamma," "max_depth," "colsample_bytree," and "min_child_weight" are used in this study, while the remaining hyperparameters are set to their default values in Python [27].

Table 1. XGBoost parameters tuning for the model

Name	Description
learning_rate	In model updates, step size shrinking is used.
subsample	The ratio of training instances that are randomly selected for fitting each individual tree.
max_leaves	The maximum number of nodes that can be added to a tree.
max_depth	The maximum depth allowed for a tree.
gamma	The minimum amount of loss reduction that is required to partition further.
colsample_bytree	Colsample_bytree: The ratio of features/columns that are randomly selected for fitting each individual tree.
min_child_weight	The minimum weight required for instances to be included in a leaf.
reg_alpha	L1 Regularization term on weights

The gamma parameter is preferred over the min_child_weight parameter because it regulates the complexity resulting from the loss, rather than the loss derivative from the hessian weight. The objective is to fit the parameters to the data without overfitting, which means tuning the algorithm to the extent that it identifies too many characteristics that are only relevant to the present data. Each parameter has its own potential for causing problems. Prior to running XGBoost, three types of parameters must be established: general parameters, booster parameters, and task parameters. The parameters used in XGBoost can be classified into three types: general parameters, booster parameters, and learning task parameters. General parameters are used to specify the type of booster being used for boosting, such as a tree or linear model. Booster parameters, on the other hand, are specific to the chosen booster and determine its behavior during training. Finally, learning task parameters, such as regression or ranking tasks, are used to determine the type of learning scenario, and many sets of parameters may be necessary. We will rigorously fine-tune the hyperparameters of these machine learning algorithms to obtain optimal intrusion detection capabilities to further improve the IDS's effectiveness. Finally, our research aims to contribute significantly to improving the overall security of computer systems and networks by effectively identifying and mitigating potential security threats such as infiltration, misuse, and anomalies.

3. METHODS

The University of New Brunswicks Canadian Institute, for Cybersecurity (CIC) has developed a dataset called CSE-CIC-IDS-2018 [28], [29]. This dataset includes 9 million records of network traffic data encompassing both normal and malicious activities. It covers a range of attack types such as denial of service (DoS), distributed denial of service (DDoS), reconnaissance, penetration, and botnet activities. The dataset is publicly available in formats like CSV files. Serves as a valuable resource for researchers and practitioners to create and test intrusion detection algorithms in a controlled lab setting. The dataset contained 16,233,002 records from a significant network comprising attack and victim workstations [28], [29]. DoS assaults, Brute force attacks, Botnet attacks, DDoS attacks, Web attacks, and infiltrations are among the 14 different types of attacks and six different infiltration scenarios included in the dataset. Brute force-web, botnet, Secure Shell (SSH) brute force, DDoS - High Orbit Ion Cannon (HOIC), DDoS - Low Orbit Ion Cannon (LOIC), user datagram protocol (UDP), and HTTP attacks, structured query language (SQL) injections, Brute force-cross-site scripting (XSS), DoS GoldenEye, DoS Hulk, DoS slow HTTP test, infiltration, and DoS Slowloris are all examples of DDoS attacks. The dataset has been widely used to develop IDS using machine learning techniques and is now the standard for anomaly-based IDS implementations. The study's focus is on the analysis and pre-processing of the CSE-CIC-IDS-2018 dataset, which consisted of ten raw data files with 16 million unique network flows representing various types of attacks. These files were combined to form a single dataset during the integration stage.

Following an examination of the abstract and an initial literature review, a problem was detected, leading to the discovery of a defect. The design of the model represented in Figure 1 was inspired by this shortcoming. It illustrates the HO-HGB algorithm implemented, starting with data preprocessing, exploratory data analysis, and data preparation in step 1. For detailed information, please refer to sections experiment design (A) and (B). After constructing the dataset, machine learning techniques, including XGBoost and traditional methods, are applied following a train-test split procedure to assess their predictive performance. The XGBoost model is trained using the input data, and hyperparameters are fine-tuned to optimize

algorithm configuration for the dataset. Finally, a performance evaluation is conducted to assess the effectiveness of the model. The proposed methodology in the development phase is validated, and system functionality is evaluated by comparing the classification results with those of the same dataset. In this paper, the labeled network flows will be divided into two categories for analysis: attacks and benign. The benign category will include all traffic classified as normal, while the attacks category will include all traffic classified as anomalous. The dataset is split into 70.1% benign traffic and 29.9% anomaly traffic.

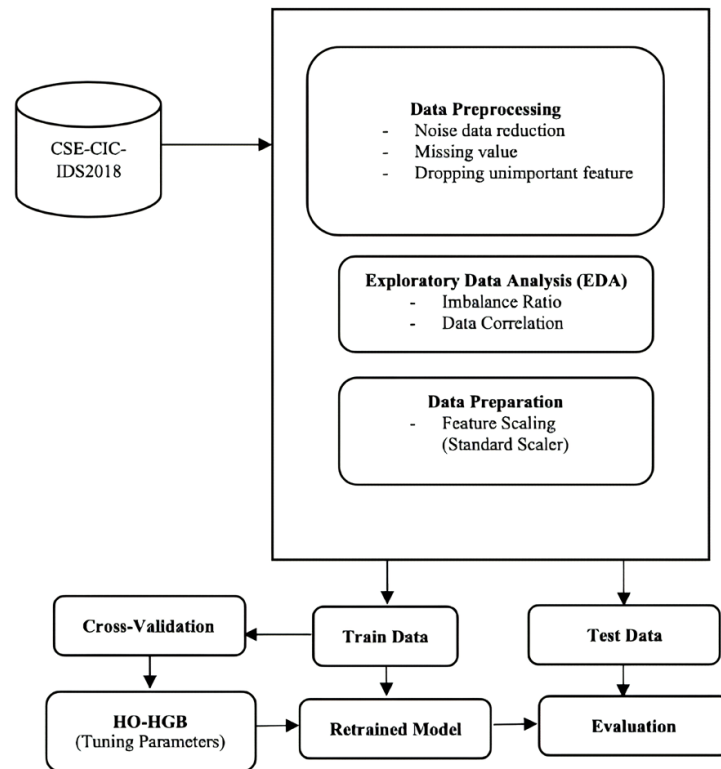


Figure 1. The flow chart of the proposed model

4. EXPERIMENTAL SETUP

The system utilized in this study was a 64-bit macOS Ventura with the following specifications: an eight-core Intel Core Xeon W processor running at 3.2 GHz, 32 GB of 2666 MHz DDR4 memory. Python version 3.11 environment was used, and the implementation and evaluation of the recommended model were carried out using NumPy [30], pandas [19], and sklearn [31] packages for data processing. Data handling, preprocessing, and analysis were performed using Pandas and NumPy libraries, while Scikit Learn was utilized for model training, evaluation, and evaluation metrics. Data visualization was carried out using the Seaborn library and Matplotlib. The following subsections provide more detailed information.

4.1. Data pre-processing

"Data pre-processing" in machine learning refers to the process of preparing the original data for use with machine learning (ML) algorithms. This involves tasks such as data cleaning, feature scaling (to standardize the range of data, particularly when there is a large variation between values among different features to avoid bias from outliers), and feature engineering. Categorical variables were encoded using one-hot encoding to convert them into binary representations.

After removing variables with missing values from the original dataset, we replaced the remaining variables' missing values. We describe the experiment settings, including how to split the dataset, address class imbalance issues, and implement seven machine learning classifiers. All dataset processing steps are fully documented here.

To make the training less sensitive to feature scaling and avoid similar sounds when applying the model to the test dataset, the sklearn preprocessing package was used to replace the majority of the data with its standard deviation and scale it to a range of 0 to 1 using the MinMaxScaler. This results in each numerical feature being set to the range of 0.0 to 1.0.

- Features with "NaN" values like "Bwd PSH Flags", "Bwd URG Flags", "Fwd URG Flags", "Fwd Byts/b Avg", "CWE Flag Count", "Fwd Pkts/b Avg", "Fwd Pkts/b Avg", and "Bwd byts/b Avg" were removed.

- Eight fields, including "Bwd PSH flags", "Bwd URG flags", "Fwd Avg Bytes Bulk", "Fwd Avg Packets Bulk", "Fwd Avg Bulk Rate", and "Bwd Avg Bytes Bulk", were eliminated from each instance.
- Negative values such as "Init Win bytes forward" and "Init Win bytes backward" were disregarded.
- The Protocol field was eliminated as it is rather redundant, given that the Dst_Port (Destination Port) field primarily contains comparable protocol values for each destination port value.
- Two columns ('Flow Bytes' and 'Flow Pkts') contained infinity values and were set to the maximum value in the column.
- The 'Timestamp' column was removed to prevent the learners from differentiating between attack predictions based on time, particularly when dealing with covert attacks.
- Attacks were assigned numerical values and are now represented in the 'Label' column.

The dataset's feature count drops from 80 to 69 after preprocessing. Then, both training and testing subsets are built using this enhanced data. With fewer features, the dataset is more streamlined, with unnecessary data being removed and efficiency being improved. Strong model training and evaluation are made possible by the development of separate training and testing subsets, guaranteeing that the model's prediction skills are properly refined. The accuracy and effectiveness of the model are optimized through this procedure, making it suitable for practical implementations in real-world circumstances. This involves concentrating on the most important qualities and eliminating the unnecessary ones.

4.2. Performance evaluation criteria

In this section, the evaluation criteria for assessing the performance of the proposed IDS are outlined. Various metrics such as accuracy (ACC), false alarm rate (FAR), and detection rate (DR) [7] were used to evaluate the IDS. The confusion matrix (CM) was used to determine the number of links correctly identified by the classifier as anomalies true positives (TP) and true negatives (TN). TN represents the number of normal connections that the classifier correctly classified as normal. In addition to TP and TN, the CM also includes false negatives (FN) and false positives (FP) to indicate the classification result. FN is the number of anomaly connections that the classifier improperly labeled as normal, whereas FP represents the number of normal connections that the classifier incorrectly tagged as anomalies. The true positive rate (TPR), also known as recall or sensitivity, is calculated as (2).

$$\text{Sensitivity} = \frac{TP}{TP+FN} \quad (2)$$

The sensitivity metric may be biased if FP and TN are excluded from the calculation, especially when dealing with unbalanced class distributions. Additionally, the following formula is used to calculate the true negative rate (TNR), also known as specificity,

$$\text{Specificity} = \frac{TN}{TN+FP} \quad (3)$$

When dealing with unbalanced data classes, specificity may offer false insights by ignoring FN and TP. The F1-score, which takes into account both recall and precision, shows to be a better evaluation tool for a more thorough analysis. It accounts for both false positives and negatives, providing a comprehensive evaluation of a model's performance that is less prone to distortion in situations when class distribution is asymmetric.

$$\text{F1 Score} = 2 * (\text{Precision} * \text{Recall}) / (\text{Precision} + \text{Recall}) \quad (4)$$

Various performance indicators, including accuracy, precision, recall, and F1-score, are used to assess the success of machine learning classification models. The F1 score is a model score that combines recall and precision. This statistic, like Accuracy, takes accuracy and recall into account when evaluating the model's performance. For machine learning models, the F-score is a valuable performance measure.

5. RESULTS AND DISCUSSIONS

The pearson correlation coefficient was used to find and exclude strongly linked variables. Our study showed 13 features with a correlation over 0.7, including 'Dst Port', 'Flow Byts/s', 'Fwd URG Flags', 'FIN Flag Cnt', 'PSH Flag Cnt', 'ACK Flag Cnt', 'URG Flag Cnt', 'CWE Flag Count', 'Init Fwd Win Byts', 'Init Bwd Win Byts', 'Fwd Seg Size Min', 'Active Std', and 'Idle Std'. To compare our model with other successful machine learning methods previously trained on this dataset, we employed logistic regression, decision tree classifier, random forest classifier, k-nearest neighbors' classifier, and XGBoost classifier. We also used hyperparameter optimization XGBoost to outperform traditional machine learning techniques. We evaluated the algorithms' performance using accuracy, precision, recall, and F1-score and generated performance

indicators using the classification report function in the Scikit Learn module in Python. The use of different classifier techniques in intrusion detection systems is an emerging study in machine learning. Table 2 presents the performances obtained from traditional Machine Learning. The metrics for each classifier have been compiled, and according to the table, the XGBoost Classifier had a positive impact on the dataset, achieving 100% success. As seen in the table, the XGBoost Classifier achieved a 99.84% success rate for the CSE-CIC-IDS2018 dataset.

Table 2. Comparison XGBoost with traditional machine learning algorithms

No.	Model Name	Train Accuracy	Test Accuracy	ROC Score	Specificity	Sensitivity	F1 Score
1	Logistic Regression	0.849186	0.849559	0.784295	0.620286	0.948305	0.8423
2	DecisionTree Classifier	1.000000	0.999660	0.999581	0.999383	0.999779	0.9997
3	Random Forest Classifier	1.000000	0.999660	0.999523	0.999178	0.999867	0.9997
4	XGBoost Classifier	1.000000	0.999938	0.999926	0.999897	0.999956	0.9999
5	KNNClassifier	0.992801	0.987997	0.987231	0.985305	0.989156	0.9880

In this experiment, we compared the performance of XGBClassifier with traditional machine learning techniques on the dataset. The results are presented in the form of the receiver operating characteristic (ROC) curve, with the performance curve on classification depicted in Figure 2. XGBoost identified as the most effective approach based on the results. Following experiments involved fine-tuning the parameters to obtain peak performance. It demonstrates that the proposed XGBClassifier outperformed traditional methods, achieving a ROC score of approximately 0.999926. The performance is considered optimal when the ROC curve approaches the upper left corner.

A set of hyperparameter setups for a machine learning model, presumably connected to XGBoost are shown in Table 3. The hyperparameters it includes are `learning_rate`, `n_estimators`, `max_depth`, `min_child_weight`, `gamma`, `subsample`, `colsample_bytree`, and `reg_alpha`. Each row in the table represents a separate parameter, while the columns reflect multiple runs or configurations (HO-XGB1 through HO-XGB7). These hyperparameters regulate key aspects of the behavior of the model, such as the step size of the learning rate, the number of boosting rounds, the depth of the tree, instance weight thresholds, regularization, and the ratios of feature and sample subsampling. To optimize the model's performance and successfully adapt it to diverse machine learning tasks, hyperparameter tweaking is necessary. This is done by experimenting with various combinations of these variables.

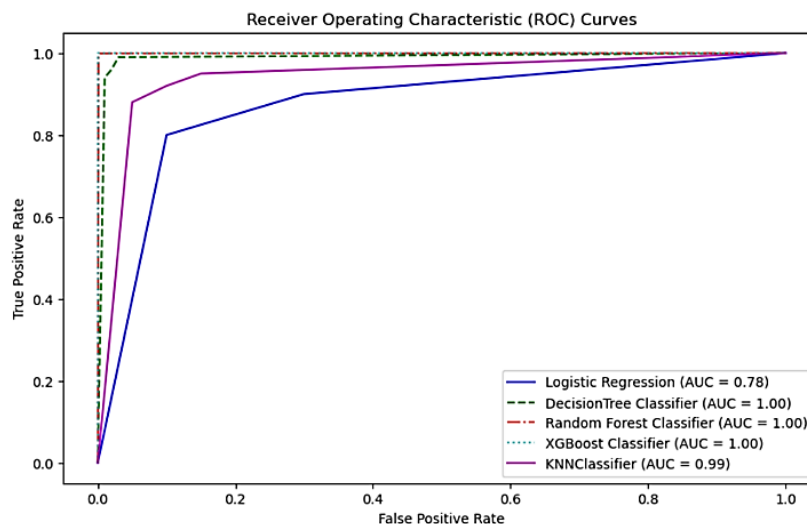


Figure 2. ROC curves of classification on CSE-CIC-IDS2018 dataset

Table 3. The various parameters of hyperparameter optimization XGBoost (HO-XGB)

Parameters	HO-XGB1	HO-XGB2	HO-XGB3	HO-XGB4	HO-XGB5	HO-XGB6	HO-XGB7
learning_rate	0.1	0.1	0.01	0.01	0.1	0.1	0.01

n_estimators	1000	1000	5000	1000	500	100	100
max_depth	5	4	4	4	10	3	6
min_child_weight	1	6	6	1	1	1	1
Gamma	0	0	0	10	10	1	1
Subsample	0.8	0.8	0.8	1	0.8	0.8	1
colsample_bytree	0.8	0.8	0.8	0.4	0.4	0.8	0.8
reg_alpha	0	0	0.005	0.3	0.3	0.3	0.3

The significance of picking optimal combinations of hyperparameters rather than painstakingly assessing each parameter individually is highlighted in this research. Several hyperparameter tuning strategies were studied in order to determine the most successful ones. Random_state = 25, nthread = 4, scale_pos_weight = 1, seed = 27, and Objective = 'binary: logistic' were also used. The results showed that the xgboost complexity was extremely limited, making it difficult to create trees without pruning because the loss threshold was not met due to Gamma. The closeness of the train and test datasets contributed to this problem. Notably, changing the parameters, notably the max_depth and min_child_weight values, resulted in significant improvements in effectiveness. Based on these findings, the suggested model, HO-XGB1, performed admirably in solving the network intrusion detection problem, as proven by an amazing ROC score of 0.999991 and an F1 score of 1.0, as shown in Table 4. The experimental results show that HO-XGB1 outperforms different parameter settings, effectively optimizing XGBoost's hyperparameters for intrusion detection on the CSE-CIC-IDS-2018 dataset.

Table 4. Comparison of the result by various hyperparameter tuning

No.	Model Name	Train Accuracy	Test Accuracy	ROC Score	Specificity	Sensitivity	F1 Score
1	HO-XGB1	1.000000	0.999988	0.999991	1.000000	0.999982	1.0000
2	HO-XGB2	0.999992	0.999988	0.999991	1.000000	0.999982	1.0000
4	HO-XGB3	0.999992	0.999988	0.999991	1.000000	0.999982	1.0000
5	HO-XGB4	0.999870	0.999852	0.999853	0.999855	0.999850	0.9999
6	HO-XGB5	0.999981	0.999975	0.999982	1.000000	0.999965	1.0000
7	HO-XGB6	0.956733	0.957333	0.965731	0.986641	0.944821	0.9580
8	HO-XGB7	0.998658	0.998719	0.998594	0.998284	0.998905	0.9987

Our ultimate goal is to create and deploy an intrusion detection system (IDS) that accurately classifies network traffic using machine learning methods. Furthermore, in the context of intrusion detection, we propose to undertake a full performance evaluation of five unique machine learning methods (logistic regression, decision tree, random forest, XGBoost, and K-neighbor). We will also concentrate on tuning the hyperparameters of these machine learning algorithms to achieve peak performance in the intrusion detection task. Finally, our research intends to greatly improve the overall security of computer systems and networks by identifying and mitigating potential security threats such as infiltration, misuse, and anomalies.

6. CONCLUSION

The goal of this research is to create a strong intrusion detection system that uses machine learning techniques and tailored hyperparameters to improve the security of our increasingly linked digital world. We intend to make a substantial contribution to the ongoing effort to secure sensitive data and networks from harmful attacks by fulfilling these goals. The proposed approach has been tested on real-world CSE-CIC-IDS2018 datasets, and the performance of XGBoost was compared to that of a traditional classification model using metrics such as accuracy, area under the ROC curve (AUC), recall, and F1 score obtained from a 10-fold cross-validation. According to the findings, XGBoost surpasses other detection algorithms. To fully exploit the benefits of XGBoost, we created the HO-XGB model, which entails fine-tuning multiple hyperparameters. We investigated the effect of learning_rate, subsample, max_leaves, max_depth, gamma, colsample_bytree, min_child_weight, n_estimators, max_depth, and reg_alpha on algorithm performance. The HO-XGB1's remarkable performance may be attributed to the careful selection and tuning of hyperparameters. The model successfully lowered complexity by improving max_depth and min_child_weight, producing excellent results with a ROC score of 0.999991 and an F1 score close to 1.0. This outperformance of various parameter settings on the CSE-CIC-IDS-2018 dataset demonstrates HO-XGB1 as the better solution for network intrusion detection. HO-XGB1 was able to successfully address the intrusion detection challenge because of the researchers' attention to hyperparameter tuning and comprehensive assessment of model complexities, making it a highly effective solution for real-world cybersecurity applications. Despite numerous machine learning techniques being proposed for intrusion

detection systems (IDS), the desired level of performance has not yet been achieved. This is because the types of network attacks have evolved, highlighting the need to update the datasets used for evaluating IDS. Moving forward, we plan to explore various clustering strategies to enhance accuracy across a range of domains.

ACKNOWLEDGEMENTS

My heartfelt gratitude goes to Suan Dusit University for providing equipment as well as scientific and technical assistance.




REFERENCES

- [1] A. Jassam Mohammed, M. Hameed Arif, and A. Adil Ali, "A multilayer perceptron artificial neural network approach for improving the accuracy of intrusion detection systems," *IAES International Journal of Artificial Intelligence (IJ-AI)*, vol. 9, no. 4, pp. 609–615, Dec. 2020, doi: 10.11591/ijai.v9.i4.pp609-615.
- [2] X. Zong, R. Li, and Z. Ye, "An Intrusion Detection Model Based on Improved Whale Optimization Algorithm and XGBoost," in *2021 11th IEEE International Conference on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications (IDAACS)*, Sep. 2021, pp. 542–547. doi: 10.1109/idaacs53288.2021.9660858.
- [3] S. Sasikumar and C. N. S. Vinoth Kumar, "Network Intrusion Detection and Deduce System," *Turkish Journal of Computer and Mathematics Education*, vol. 12, no. 9, pp. 404–410, 2021.
- [4] S. Sharipuddin *et al.*, "Intrusion detection with deep learning on internet of things heterogeneous network," *IAES International Journal of Artificial Intelligence (IJ-AI)*, vol. 10, no. 3, pp. 735–742, Sep. 2021, doi: 10.11591/ijai.v10.i3.pp735-742.
- [5] S. Chimphlee and W. Chimphlee, "Machine learning to improve the performance of anomaly-based network intrusion detection in big data," *Indonesian Journal of Electrical Engineering and Computer Science*, vol. 30, no. 2, pp. 1106–1119, May 2023, doi: 10.11591/ijeecs.v30.i2.pp1106-1119.
- [6] L. N. Tidjon, M. Frappier, and A. Mammar, "Intrusion Detection Systems: A Cross-Domain Overview," *IEEE Communications Surveys & Tutorials*, vol. 21, no. 4, pp. 3639–3681, 2019, doi: 10.1109/comst.2019.2922584.
- [7] O. Sbai and M. Elbouchari, "Deep learning intrusion detection system for mobile ad hoc networks against flooding attacks," *IAES International Journal of Artificial Intelligence (IJ-AI)*, vol. 11, no. 3, pp. 878–885, Sep. 2022, doi: 10.11591/ijai.v11.i3.pp878-885.
- [8] Yasir Hamid, Balasaraswathi Ranganathan, Ludovic Journaux, and Sugumaran Muthukumarasamy, "Benchmark Datasets for Network Intrusion Detection: A Review," *International Journal of Network Security*, vol. 20, no. 4, pp. 645–654, 2018.
- [9] S. M. Othman, N. T. Alsohybe, F. Mutaheer Ba-Alwi, and A. T. Zahary, "Survey on Intrusion Detection System Types," *International Journal of Cyber-Security and Digital Forensics*, vol. 7, no. 4, pp. 444–462, 2018.
- [10] S. M. Othman, F. M. Ba-Alwi, N. T. Alsohybe, and A. Y. Al-Hashida, "Intrusion detection model using machine learning algorithm on Big Data environment," *Journal of Big Data*, vol. 5, no. 1, pp. 1–12, Sep. 2018, doi: 10.1186/s40537-018-0145-4.
- [11] M. Ring, S. Wunderlich, D. Scheuring, D. Landes, and A. Hotho, "A survey of network-based intrusion detection data sets," *Computers & Security*, vol. 86, pp. 147–167, Sep. 2019, doi: 10.1016/j.cose.2019.06.005.
- [12] Y. Ayachi, Y. Mellah, M. Saber, N. Rahmoun, I. Kerrakchou, and T. Bouchentouf, "A survey and analysis of intrusion detection models based on information security and object technology-cloud intrusion dataset," *IAES International Journal of Artificial Intelligence (IJ-AI)*, vol. 11, no. 4, pp. 1607–1614, Dec. 2022, doi: 10.11591/ijai.v11.i4.pp1607-1614.
- [13] M. Verkerken, L. D'hooge, T. Wauters, B. Volckaert, and F. De Turck, "Towards Model Generalization for Intrusion Detection: Unsupervised Machine Learning Techniques," *Journal of Network and Systems Management*, vol. 30, no. 1, Oct. 2021, doi: 10.1007/s10922-021-09615-7.
- [14] P. R. Chandre, P. Mahalle, and G. Shinde, "Intrusion prevention system using convolutional neural network for wireless sensor network," *IAES International Journal of Artificial Intelligence (IJ-AI)*, vol. 11, no. 2, pp. 504–515, Jun. 2022, doi: 10.11591/ijai.v11.i2.pp504-515.
- [15] T. Xia, "Support Vector Machine Based Educational Resources Classification," *International Journal of Information and Education Technology*, vol. 6, no. 11, pp. 880–883, 2016, doi: 10.7763/ijiet.2016.v6.809.
- [16] H. Alqahtani, I. H. Sarker, A. Kalim, S. M. Minhaz Hossain, S. Ikhlaiq, and S. Hossain, "Cyber Intrusion Detection Using Machine Learning Classification Techniques," in *Computing Science, Communication and Security*, Springer Singapore, 2020, pp. 121–131. doi: 10.1007/978-981-15-6648-6_10.
- [17] Saroj Kr. Biswas, "Intrusion Detection Using Machine Learning: A Comparison Study," *International Journal of Pure and Applied Mathematics*, vol. 118, no. 2018, pp. 101–114, 2018, [Online]. Available: <http://www.ijpam.eu>
- [18] W. L. Al-Yaseen, Z. A. Othman, and M. Z. A. Nazri, "Multi-level hybrid support vector machine and extreme learning machine based on modified K-means for intrusion detection system," *Expert Systems with Applications*, vol. 67, pp. 296–303, Jan. 2017, doi: 10.1016/j.eswa.2016.09.041.
- [19] K.-A. Tait *et al.*, "Intrusion Detection using Machine Learning Techniques: An Experimental Comparison," in *2021 International Congress of Advanced Technology and Engineering (ICOTEN)*, Jul. 2021, pp. 1–10. doi: 10.1109/icoten52080.2021.9493543.
- [20] S. P. Kumar and A. Raaza, "Study and analysis of intrusion detection system using random forest and linear regression," *Periodicals of Engineering and Natural Sciences (PEN)*, vol. 6, no. 1, pp. 197–200, Jun. 2018, doi: 10.21533/pen.v6i1.289.
- [21] A. Ammar, "A Decision Tree Classifier for Intrusion Detection Priority Tagging," *Journal of Computer and Communications*, vol. 03, no. 04, pp. 52–58, 2015, doi: 10.4236/jcc.2015.34006.
- [22] D. Zhang, L. Qian, B. Mao, C. Huang, B. Huang, and Y. Si, "A Data-Driven Design for Fault Detection of Wind Turbines Using Random Forests and XGboost," *IEEE Access*, vol. 6, pp. 21020–21031, 2018, doi: 10.1109/access.2018.2818678.
- [23] H. Li, H. Jiang, D. Wang, and B. Han, "An Improved KNN Algorithm for Text Classification," in *2018 Eighth International Conference on Instrumentation & Measurement, Computer, Communication and Control (IMCCC)*, Jul. 2018, pp. 1081–1085. doi: 10.1109/imccc.2018.00225.
- [24] T. Chen and C. Guestrin, "XGBoost: A Scalable Tree Boosting System," in *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Aug. 2016, pp. 785–794. doi: 10.1145/2939672.2939785.
- [25] S. Dhaliwal, A.-A. Nahid, and R. Abbas, "Effective Intrusion Detection System Using XGBoost," *Information*, vol. 9, no. 7, p. 149, Jun. 2018, doi: 10.3390/info9070149.




- [26] O. E. Ørebæk and M. Geitle, “Exploring the hyperparameters of XGBoost through 3D visualizations,” 2021.
- [27] S. Mukhopadhyay, *Advanced Data Analytics Using Python*. United Kingdom: Apress Berkeley, CA, 2018. doi: 10.1007/978-1-4842-3450-1.
- [28] “CSE-CIC-IDS2018 on AWS,” *Canadian Institute for Cybersecurity*. 2018. [Online]. Available: <https://www.unb.ca/cic/datasets/ids-2018.html>
- [29] J. L. Leevy and T. M. Khoshgoftaar, “A survey and analysis of intrusion detection models based on CSE-CIC-IDS2018 Big Data,” *Journal of Big Data*, vol. 7, no. 1, pp. 1–19, Nov. 2020, doi: 10.1186/s40537-020-00382-x.
- [30] M. A. Ferrag, L. Maglaras, S. Moschogiannis, and H. Janicke, “Deep learning for cyber security intrusion detection: Approaches, datasets, and comparative study,” *Journal of Information Security and Applications*, vol. 50, p. 102419, Feb. 2020, doi: 10.1016/j.jisa.2019.102419.
- [31] P. P. Iouliauou, V. G. Vassilakis, and S. F. Shahandashti, “A Trust-Based Intrusion Detection System for RPL Networks: Detecting a Combination of Rank and Blackhole Attacks,” *Journal of Cybersecurity and Privacy*, vol. 2, no. 1, pp. 124–152, Mar. 2022, doi: 10.3390/jcp2010009.

BIOGRAPHIES OF AUTHORS



Witcha Chimphee    holds a Ph.D. in Computer Science from University Technology of Malaysia. He is currently an Assistant Professor in the Data Science and Analytics department at Suan Dusit University, Thailand. Computer Networks and Security, Machine Learning, Data science, and big data analytics are among his current research interests. Witcha has a long track record of publication in peer-reviewed publications and has actively participated in a number of international conferences. He is a determined researcher who is dedicated to furthering the subject of computer science. He can be contacted at email: witcha.chi@gmail.com



Siriporn Chimphee    holds a Ph.D. in Computer Science from the University Technology of Malaysia and is currently an Assistant Professor in Suan Dusit University, Thailand. Data mining, Intrusion Detection, web mining, and information technology are among her research interests. Dr. Siriporn has a significant publication record and participates in international conferences on a regular basis. She is really enthusiastic about exploring new frontiers in computer science and is constantly looking for ways to progress the subject through her research. She can be contacted at email: siriporn.chi@gmail.com